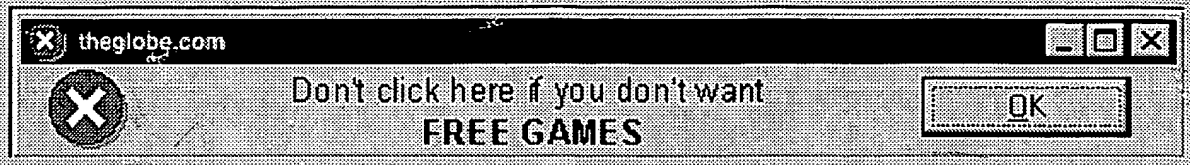

ActiveX Programming Unleashed

Appendix F - HTML Enhancement by Internet Explorer 3.0

ACTIVEX PROGRAMMING UNLEASHED, Second Edition
(Imprint: SAMS.Net)
Publication Date: Dec-96
Author: Weiying Chen
ISBN: 1-57521-154-8

Chapter 1

An Overview of ActiveX

by Weiying Chen

Microsoft has unveiled an extensive new solution technology for the Internet called *ActiveX.* Microsoft ActiveX is a broad and powerful abstraction for Microsoft Internet Solutions. Content providers and Internet application developers now have a robust and extensible frameworks that enables them to develop a new generation of Internet applications.

Microsoft is aggressively adding features to the Win32 *Application Programming Interfaces* (*APIs*) that will let developers "Internet-enable" their applications. These new features are based on *Object Linking and Embedding* (*OLE*) technology so that developers who have made investments into Win32 and OLE applications can leverage their investments.

ActiveX exposes a set of APIs that enables developing a new generation of client/server applications for the Internet. ActiveX has interfaces to integrate almost every media technology within an application. It provides extensive support for animation, 3D virtual reality, real-time audio, and real-time video.

ActiveX gives developers an open framework for building innovative applications for the Internet. ActiveX technologies form a robust framework for creating interactive content using reusable components, scripts, and existing applications. Specifically, ActiveX technologies enable content providers and application developers to create powerful and dynamic Web content and Web server extensions quite easily. This feat is achieved by using ActiveX controls, Active client and server side scripts, and the Active document interfaces and ISAPI (Internet Server Application Programming Interface).

COM: The Fundamental "Object Model" for ActiveX and OLE

COM (Component Object Model) is the technical cornerstone for the ActiveX technology; it defines how objects expose themselves for use within other objects and how objects can communicate between processes and across a network. You can easily integrate COM objects for use in many languages, such as Java, Basic, and C++. COM *objects* are reusable binary components.

The following concepts are fundamental to COM:

- Interface: The mechanism through which an object exposes itself.
- IUnknown Interface: The interface on which all others are based. It implements the reference-counting and interface-querying mechanisms required for COM objects.
- Reference Counting: The technique by which an object keeps track of its reference instance count. The instance of the object class should be deleted when there is no reference to this instance.
- QueryInterface Method: It is called with the Interface ID (IID) to which the caller wants a pointer. can be generated by Guidgen.exe by choosing DEFINE_GUID(...) format. QueryInterface enables navigation to other interfaces exposed by the object.
- IClassFactory Interface: This interface must be implemented for every object class. It provides functionality to create an instance of the object class with CLSID and locks the object server in memory to allow creation of objects more quickly.
- Marshaling: The mechanism that enables objects to be used across process and network boundaries, allowing interface parameters for location independence by packing and sending them across the process boundary. Developers have to create proxy/stub dll for the custom interfaces if exist. The custom interface has to be registered in the system registry.
- Aggregation: COM object supports an interface by including another object that supports that interface. The containing object creates the contained object as part of its own creation. The result is that the containing object exports the interface for the contained object by not implementing that interface.
- Multiple Inheritance: A derived class may inherit from multiple interfaces.

ActiveX Object Model

There are two primary pieces to the ActiveX Object Model: the Microsoft HyperText Markup Language (HTML) Viewer component (MSHTML.dll) object and the Web Browser Control (shdocvw.dll). Both are in-process (DLL-based) COM objects.classes.

All interfaces defined in the ActiveX Object Model are "dual" interfaces. A "dual" interface means that the objects inherit from IDispatch and IUnknown. They can be used by client application at "early-bind" via Vtable and at "late bind" via OLE automation controller by using IDispatch::GetIdsOfNames and IDispatch::Invoke.vtable).

MSHTML is the HTML viewer part of Microsoft Internet Explorer 3.0. It is an in-process COM server and a Document Object. It can be hosted in OLE Document Object containers.

MSHTML implements the OLE Automation object model described in the HTML Scripting Object Model. With this object model, you can develop rich multimedia HTML content. VBScript running inline in the HTML and Visual Basic 4.0 running external to the HTML can use the object model.

The Web browser control object is an in-process COM Server. It also serves as a Document Object container that can host any Document Objects, including MSHTML, with the added benefit of fully supporting hyperlinking to any document type.

The Web browser control is also an OLE control. The IWebBrowser interface is the primary interface exposed by the Web Browser Control.

The Web browser control is the core of what customers see as "the Internet Explorer 3.0 product.". Internet Explorer 3.0 also provides a frame to host this control. Internet Explorer 3.0 supports the following HTML 3.x0 extensions:

- Frame: Creates permanent panes for displaying information, supporting floating frames or borderless frames..
- NOFRAMES: Content that can be viewed by browsers not supporting frames
- OBJECT: Inserts an OLE control
- TABLE: Fully compliant with HTML 3.x0 tables with cell shading and text wrapping.
- StyleSheet: font size, intra-line space, margin, highlighting and other features related with styles can be specified in the HTML by the user.
- In-Line sound and video

ActiveX Controls

ActiveX Control (formerly known as OLE control) has a broader definition. It refers to any COM objects. For instance, the following objects are all considered an ActiveX control.

- Objects that expose a custom interface and the IUnknown interface
- OLE automation servers that expose the IDispatch/Dual interfaces
- Existing OLE controls (OCX)
- OLE objects that make use of monikers
- Java Applet with the support of COM

Note

A moniker acts as a name that uniquely identifies a COM object. It is a perfect programming abstraction for Uniform resource locator(URL).

ActiveX Control used inside scripting languages make this binary reusable components reused in the Internet world. Almost any type of media wrapped into an ActiveX control can be seamlessly integrated into your Web page. Sound, video, animation, or even credit-card approvals controls can be used within your Web page.

ActiveX Scripting

ActiveX Scripting is the interface for script engines and script hosts. Following this interface specification, the script vendors can use their custom script engine in any script host, such as IE 3.0. What's more, with its infrastructure, developers can choose any script language they prefer. A *script* is some executable block, such as a DOS batch file, Perl script or an EXE file.

ActiveX Scripting components can be grouped into two major categories: an ActiveX Scripting Engine and an ActiveX Scripting host. A *host* creates a script engine so that scripts can run against the host.

Here are some examples of ActiveX Scripting hosts:

- Microsoft Internet Explorer
- Internet Authoring tools
- Shell

The ActiveX Scripting Engine is an OLE COM object that supports the IOLEScript interfaces, with at least one of the IPersist interfaces and an optional IOleScriptParse interface.

ActiveX Scripting Engines can be developed for any language, such as

- Microsoft Visual Basic Script Edition (VBScript)
- JavaScript
- Perl

Microsoft ActiveX Scripting Languages products, such as Visual Basic Script and Java Script, can be used to "glue" together the functionality exposed in ActiveX Controls to create rich

Web-based applications.

Active Documents

Active Documents are based on the OLE Document Objects (DocObjects, for short). The DocObjects technology is a set of extensions to OLE Compound Document technology. It is the core technology that makes Microsoft Office Binder work. Active Document also refers to any document that contains ActiveX controls, a Java Applet, or a Document Object.

One other obvious application for this technology is "Internet browsers." You can open richly formatted documents, such as Microsoft Word and Excel spreadsheets, directly in the browser.

Figure 1.1 shows how seamlessly the Word document placed in IE 3.0. The word *toolbar* is added to the browser, enabling you to work on the document while cruising the Internet.

Figure 1.1. Active document.

The following describes the general criteria for the document object container and the document object.

A Document Object container must implement the following objects and interfaces:

- Document Site objects with IOleClientSite, IAdviseSink, and IOleDocumentSite interface exposed.
- View Site objects with IOleInPlaceSite and IContinueCallBack interface exposed.
- Frame Objects with IOleIPlaceFrame and IOleCommandTarget interface exposed
- IOleDocumentSite to support the DocObjects
- IOleCommandTarget on the frame object
- IOleClientSite and IAdvisesink for "site" object.
- IPersistStorage to handle object storage
- IOleInPlaceSite to support in-place activation of embedded object
- IOleInPlaceFrame for container's frame object

Similarly, Document Objects must implement the following objects and interfaces:

Objects

- Document Object with IDataObject, IPersistStorage, IPersistFile, and IOleDocument interface exposed.
- View Object with IOleInPlaceObject, IOleInPlaceActiveObject, IOleDocumentView, IPrint, and IOleCommandTarget interface exposed.

Containers

- IPersistStorage for storage mechanism
- IOleInPlaceObject and IOleInPlaceActiveObject for in-place activation extension of OLE documents.

- IPersistFile, IOleObject, and IDataObect to support the basic embedding features of OLE documents.
- IOleDocument, IOleDocumentView, IOleCommandTarget, and IPrint to support the Document Objects extensions interface.

For more information on the Document Object, please refer to Chapter 19, "OLE Document Objects," in this book.

ActiveX Server and Server-Side Scripting Framework

The *ActiveX Server Framework*, another component of ActiveX technologies, is based on the Microsoft Internet Information Server (IIS). IIS is built on the Windows NT Advanced Server 3.51 or greater. This framework enables developers to take advantage of the powerful Microsoft Back Office family of products, such as Microsoft SQL, SNA, Management, and Exchange Server.

Server support consists of ActiveX Server-side scripting and the usage of Aside, Batch, or JavaScript.

The *Common Gateway Interface (CGI)* is also supported under the ActiveX Server Framework. Common Gateway Interface is a protocol used to communicate between your HTML forms and your program so that your program can extract the information from the form. A lot of languages can be used to write your program, as long as the language has the capability to read the STDIN, write to the STDOUT, and read the environment variables.

An HTTP server responds to a CGI execution request from a client browser by creating a new process, reading the input from the form through the environment variable, doing some processing with the form data and write the HTML response to the STDOUT.

The server creates one process for each CGI request received. However, creating a process for every request is time consuming and takes a lot of server resources. Using too many server resources can starve the server itself.

One way to avoid this is to convert the current CGI executable file into a Dynamic Link Library(DLL) so that the DLL can be loaded into the same address space as the server. The server can load the DLL the first time it gets a request. The DLL then stays in memory, ready to service other requests until the server decides it is no longer needed. There are two types of DLLs that can be created for this purpose: one is the Internet Server application (ISA), the other is the ISAPI filter.

An ISA(Internet server application) is also known as the Internet Server Extension. There are two entry points for this DLL, GetExtensionVersion and HttpExtensionProc.

HTTP Server first calls the ISA at the entry point of GetExtensionVersion to retrieve the version number of the ISAPI specification on which the extension is based. For every client

request, the HttpExtensionProc entry point is called. Interaction between an HTTP server and an ISA is done through extension control blocks (ECBs). The ISA must be multithread-safe because multiple requests can be received simultaneously.

An ISAPI filter is a replaceable DLL that sits on the HTTP server to filter data traveling between web browser and HTTL server. Figure 1.2 shows the relationship between the HTTP server, ISA, and ISAPI filter.

Figure 1.2. HTTP server, ISA, and ISAPI filter.

ISAPI filter has the same entry point as the ISA. When the filter is loaded, it tells the server what sort of notifications it will accept. After that, whenever a selected event occurs, the filter is called and is given the opportunity to process the event.

ISAPI filters are very useful to provide the following functions:

- Authentication
- Compression
- Encryption
- Logging of HTTP requests

You can also install Multiple filters can be installed on the server. The notification order is based on the priority specified by the filter, then the load order in the registry, the registry key for filter is under \HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\parameters\Filter DLLs\. After you install the filter, IIS must be restarted.

To ease the development of the server-side application, Microsoft also provides an Internet Personalization System (IPS) on top of the IIS. It provides a server side VBScript and a set of server components and a system installed with IIS on a web server computer to make the server-side scripting available. With this system installed, you can integrate Visual Basic Script commands and expressions into the HTML files, and calls out to server side OLE automation objects. When you point your browser to a script file, this system will process the script and return HTML to the browser. This allows developers to create more elaborate content with less effort. Server-side scripting will make the server-side development a piece of cake.

Active Animation and Movies

Active Animation (formally known as Active VRML; VRML stands for Virtual Reality Modeling Language) is an advanced markup language and a viewer for 3D multimedia. The Active Animation viewer is an ActiveX control and can be used inside client side VBScript in Microsoft Internet Explorer 3.0. Web pages that use Active VRML can include interactive 2D and 3D animation, accompanied by synchronized sounds. These effects can also be triggered by events from VBScript in IE 3.0 and from other scripting languages such as JavaScript.

Active Animation synchronizes all media interaction without developers needing to write low-level code. It supports the operation and media types such as 3D geometry, images, sound,

montages, 2D and 3D points, and vectors, 2D and 3D transforms, colors, numbers, text, strings, characters, and so forth.

- Animation: The following is a list of the operations and media types supported by Active Animation:
- Sound: Supports for importing, mixing, and rendering 3D models of sound waves.
- Images: Support infinite resolutions, 2D transformation, overlaying, and rendering images from 3D models.
- 3D geometry: Supports texture mapping of animated images combined with control of color and opacity, aggregation, and transformations.
- Advanced 2D and 3D coordinate systems and transformations: Supports many advanced manipulations of vector/vector, point/vector, and scalar/vector. Also supports construction and destruction of rectangular or polar/spherical coordinates. Translate, scale, rotate, shear, inversion, identity, composition and matrix-based constructions.
- Montages: Supports multi-layered cel animation and 2 ½-D images.

Note

- The word *cel* is synonymous with a frame of animation.
- Text: Supports formatted text, its colors and its fonts families including optional bold and italics.
- Colors: Construction and destruction in RGB and HSL colors.

The Active Movie is based on the Microsoft Active Movie Streaming Format (.ASF), a new data-independent format for storing and transmitting multimedia content across the Internet. Because .ASF files can be streamed, you can begin playback of these files immediately. Active Movie is an ActiveX control and can be used inside client-side VBScript or JavaScript.

.asf is an open and extendible data-independent format. With this format, you can combine and store different data objects such as audio objects, video objects, URLs, and HTML pages into a single synchronized multimedia stream. This encapsulation feature enables popular media types and formats, such as MPEG, .avi, .wav, and Apple QuickTime, to be synchronized and stored efficiently on a variety of servers.

.asf data is also network independent and can be transmitted over different protocols and networks, including TCP/IP, UDP, RTP, IPX/SPX, and ATM.

.asf also provides efficient packaging for different network transports, supports multiple-bit rates, error correction, and other multimedia-content storage and transmissions.

You can efficiently play back .asf content by using Active Movie, Microsoft's next generation, cross-platform video technology for the desktop.

Why Use ActiveX?

With ActiveX, you can make the most of Internet resources with less effort. ActiveX Controls and Scripting give you the infrastructure needed to add language- and tool-independent

extensions to Web pages. Using ActiveX Controls lets developers take advantage of existing OLE development tools and the investment they have already made in OLE. ActiveX Scripting allows you to drop any scripting engine into IE 3.0, enabling developers to add behavior to Web pages in whatever scripting language they prefer.

ActiveX has also greatly improved extending the HTTP and FTP protocols. The ActiveX IBindXXX interfaces encapsulate a new protocol that supports the concept of binding to a URL dynamically from within your application. An application binds to a URL moniker, which then communicates through the appropriate protocol to activate the OLE object. This abstraction allows newly developed protocols to integrate into your existing objects, independent of your object design or expressed functionality.

Using the Internet Extensions for the Win32 API (WinINet) makes it easy for developers to add Internet access to their applications. WinINet abstracts the TCP/IP protocol-specific details and gives developers a simplified programming interface to access the internet instead of worrying about the WinSocket details. This API includes HTTP, FTP and Gopher access.

As with most systems, server efficiency and resource use becomes a concern when designing multi-user server applications for the Internet. The Internet Information Server offers a high-performance, secure, and extendible framework. An ISAPI Internet Server Applications (ISA) is a dynamic link library that loads into the same address space as the HTTP Server component versus CGI creates a separate process for every work request. Each new process in the CGI model requires more server resources. The advantage of developing an ISA rather than a CGI is high-level performance that requires considerably fewer resources, which frees up resources that can then be used by the server.

The Internet Database Connector allows ODBC database access through an HTTP request.

ISAPI filters can be used to enhance the Microsoft Internet Information Server with custom features, such as enhanced logging of HTTP requests, custom encryption, aliases for URLs, compression schemes, or new authentication methods. The filter applications sit between the client network connection to the HTTP server.

IIS has a few built-in ISAPI DLL. One of them is Httpodbc.dll which is called the Internet Database Connector. The Internet Database Connector allows ODBC database access through an HTTP request. Developers can use this feature to create Web pages with information from the database so that they can retrieve, insert, update, and delete information in the database based on user input and perform any other SQL commands.

Active Movie provides next-generation, cross-platform video; the Active Movie Stream Format solves several important synchronization issues in multimedia-content storage and transmission.

Active Animation gives you a powerful foundation for Internet interactive, animated, and multimedia-based content, targeting domains such as advertising, entertainment, online shopping, and technical illustration.

Microsoft is building an infrastructure around the client/server model that enables secure transactions, billing, and user authentication. IE 2.0 and 3.0 supports the secure socket layer (SSL) 2.0 and 3.0 version and personal communications technology (PCT) 1.0 version. Most importantly, ActiveX is built on Win32 and OLE, which enables developers to build on their

existing investments. ActiveX is the doorway to a whole new world of Internet applications.

ActiveX Program Development

The most fundamental model that you should understand before embarking on ActiveX development is the COM (Component Object Model). This model is the same model as discussed in the OLE COM specification.

COM is the "Object Model" for ActiveX and OLE. Microsoft also provides OLE COM wizard and ActiveX Template Library (ATL) so that developers can develop lightweight, fast COM objects. ATL offers a template to write OLE automation server, OLE controls, and the very basic dual interface or any arbitrary COM objects. ATL also provides a custom AppWizard (called OLE COM AppWizard in VC 4.x project workspace) which can be used with Visual C++ 4.1 or later to create a COM object skeleton.

Microsoft Visual C++ 4.1 Development Studio integrates different AppWizards and Control Wizard to simplify the development process. Along with Visual C++ 4.x is version 4.x of the Microsoft Foundation Classes. 4.1. In Microsoft Visual C++ 4.2, there is new support for ActiveX programming, such as

- WinInet
- Active document
- Asynchronous moniker
- URL moniker
- Control for Internet

Besides the tools, wizards, frameworks, and foundation classes, Microsoft also provides a set of specifications to implement certain ActiveX controls. For instance, it provides the OLE controls for Internet, Document object, ActiveX scripting interface, Hyperlink interface, and Asynchronous Moniker specification.

ActiveX controls can be easily manipulated by any scripting languages in IE 3.0. The scripting languages include Java script and the client side VBScript and JavaScript, or any other third party scripting language that implements the ActiveX scripting interface. ActiveX controls, particularly OLE automation servers, can also be used with the server side VBScript.

Microsoft also provides J++ to develop the Java Applet and Java Applications. Java Applet can be used as an ActiveX controls.

Besides these ActiveX client side script, Microsoft also provides a system on top of the IIS 2.0 to use the server-side VBScript. ActiveX controls, particularly OLE automation server, can be referenced in the server-side VBScript.

A variety of tools can be used to develop server-side components, such as Perl and C for CGI programming.

To facilitate developing ISAPI applications, Microsoft Visual C++ 4.x provides the ISAPI

Extension Wizard and some foundation classes for ISAPI. Along with this, Microsoft also provides the ISAPI specification.

The Active Movie add-on tool kit includes tools to develop applications that handle streamed media. This tool kit allows software developers to integrate real-time audio and video content in virtually any type of application.

Active Animation and Active Movie controls can be manipulated through client-side VBScript. A developer can glue the control's functions together without the need for complex stream synchronization methods.

Summary

ActiveX is a technology that has the potential to change the way information is accessed and used on the Internet. Powerful abstractions based on OLE have been developed that enable fast, scaleable integration of your objects within the Internet. Microsoft is making a major effort to make the Internet everything it can possibly be. By using ActiveX, developers can make the best use of their system resources while providing instant, dynamic content and functionality in their Internet applications. How information is presented greatly affects how interesting and usable people find it.

Chapter 9

JavaScript

by Rob McGregor

The Web is alive with interactivity these days, and one of the reasons for this is JavaScript. Although ActiveX components are the new kids on the block in bringing interactive content to the Web, JavaScript has already been around the block a few times. JavaScript has the capability to tie together Java applets and ActiveX components with highly interactive Web pages, providing an exciting Web browsing experience.

What Is JavaScript?

JavaScript is a scripting language that enables you to write scripts embedded in HTML pages, and it has features similar to VBScript (which was discussed in Chapter 8, "VBScript"). There are many parallels between the two scripting languages, but the underlying model is quite different. Before looking at just what JavaScript is and how it works, take a look at where it came from.

Origins of JavaScript

What is now known as JavaScript began life as a Netscape scripting language called LiveScript, originally designed for use with the Netscape LiveWire server software. The original idea was that network administrators could use LiveScript on both the server side and the client side. On the server side, administrators would control database connectivity, manage complex web sites, and automate many otherwise tedious administrative tasks. On the client side, Web page authors would be able to provide new and exciting interactive content in HTML code.

The Java Connection

LiveScript was also designed to enable HTML authors to communicate with and control Java applets. Java is a programming language derived from C and C++, yet without many of the shortcomings inherent in these languages. Java provides a fully object-oriented programming model that has built-in security features that are well suited for writing applications and applets for the Internet. The idea was that the combination of Java applets and LiveScript would allow a much richer, much more immersive Web experience for the user.

Enter Sun Microsystems

In cooperation with Sun Microsystems, Netscape hammered the bugs out of LiveScript and folded in more Java-like syntax to their new scripting language. This resulted in the December, 1995, announcement that LiveScript would henceforth be known as JavaScript.

Note

Microsoft Internet Explorer 3.0 supports an implementation of JavaScript, known as JScript, through the Microsoft dynamic link library JSCRIPT.DLL.

Differences Between JavaScript and Java

It's important to realize that Java and JavaScript are two completely separate things. Although there is a similarity in syntax between the two, they are very different. Some important differences between JavaScript and Java are listed following:

- Unlike Java, JavaScript isn't a compiled language; it's interpreted by the client's browser.
- JavaScript uses code that's embedded right into an HTML page, whereas Java applets are compiled separately but can be accessed from within HTML pages.
- JavaScript uses loosely typed, non-declared variable data types, while Java uses strongly typed, declared variable data types.
- JavaScript is object-based, whereas Java is object-oriented. This means that JavaScript code uses built-in, extensible objects, but the notions of classes and inheritance don't exist. Java, on the other hand, uses classes and inheritance in true object-oriented fashion.
- JavaScript checks object references at runtime with dynamic binding, whereas Java objects must exist at compile time and use static binding.

Like all scripting languages, JavaScript has its own syntax, and the next section examines this syntax.

JavaScript Objects

JavaScript borrows much of its syntax from Java, and Java borrows much of its syntax from C. So if you're familiar with Java, C, or C++, JavaScript should look pretty familiar, and it will probably be easier for you to use than VBScript.

A software object typically wraps some concept or functionality into an easy-to-use package. JavaScript, much like Java, expresses everything in terms of objects. An object typically contains *properties*, which define visual characteristics, and *methods*, which define functionality. The JavaScript language is composed of a rather simple hierarchy of two basic types of objects.

- Browser objects, which appear as visible user-interface objects in a browser window.
- Data objects, which are built-in data objects that remain behind the scenes. These objects represent and control certain types of data used by a Web page.

When you load an HTML page into a browser, several objects are created that correspond to the page and its contents. A page *always* has the following objects by default:

- Window
- Location
- History
- Document

In addition to these default objects, your script can specify objects that are created automatically for use by the script when the page loads. The next section explores each of the basic JavaScript objects in greater detail.

JavaScript Events

Events occur in JavaScript in response to user actions in the browser window, and these events are available for various browser objects (described in the next section). The events defined for JavaScript objects are listed in Table 9.1.

Table 9.1. Events defined for JavaScript.

Event Handler Called Just Before onBlur Input focus is removed from a form element. onClick A form element or link is clicked. onChange The value for a text, textarea, or select element has changed. onFocus A form element gets the input focus. onLoad A page is loading in the browser. onMouseOver The mouse pointer moves over a link or anchor. onSelect A form element's input field is selected. onSubmit A form's data is submitted. onUnload A browser page is exited.

Browser Objects

As stated previously, JavaScript browser objects are those that are visible in a browser window. There are five main visible objects that you'll spend most of your time working with in JavaScript, and these are as follows:

- Window object
- Document object
- Form object
- Button object
- Text object

There are many other objects in the hierarchy in addition to the five listed here. These other objects further refine the functionality of the main five, or provide ancillary services that enhance one of the main five objects.

Note

JavaScript uses an *object-based hierarchy*, not a true *object-oriented hierarchy* in the sense of Java or C++. JavaScript uses a simple containment hierarchy, so there can be no inheritance of properties or methods from parent to child as there is in Java and C++.

Take a closer look at each of the browser objects in the hierarchy to see what properties, methods, and events they provide.

Window Object

The Window object is found at the top of the JavaScript hierarchy, and everything occurs within the context of the window. The window contains properties that apply to the entire window, and it holds the document that runs the JavaScript.

Note

There can be more than one window contained within the main window. Each child window in a frames document has a corresponding window object. Frames are discussed in the "Frame Object" section later in this chapter.

Window Object Event Handlers

The window object contains two event handlers that can trigger some action in your scripts: onLoad and onUnload.

- The onLoad handler is triggered when an HTML page is loaded into the browser window.
- The onUnload event is triggered when a user leaves the page.

The onLoad and onUnload event handlers are used within either an HTML <BODY> tag or an HTML <FRAMESET> tag. For example, in a BODY tag these handlers might look like this:

```
<BODY onLoad="..." onUnload="...">
```

The "..." would be replaced by calls to some JavaScript functions (see the section "JavaScript Functions" later in this chapter for more information).

Window Object Properties

The window object contains several properties for use in your scripts, and Table 9.2 lists these properties.

Table 9.2. Window object properties.

Property Description defaultStatus The default string that appears in the window's status bar. frames An array depicting independently scrollable child windows (frames), each with a distinct URL. length The number of frames in a parent window. name The name of a window (represented by the windowName argument). parent A reference to the parent of a frames window. self An implicit parameter passed whenever a method is called, or a property is set, that refers to the same object as a given window object (similar to Me in Visual Basic or this in C++). status A string that appears temporarily in a window's status bar. top A reference to the top (main) window in a frames window relationship. window An expression referring to the windowName argument that refers to the current window.

A window object also contains these other objects as properties:

- document
- frame
- location

Although the defaultStatus property can be set at any time, it's typically done when a page loads. In this case, it's expressed as part of the window.onLoad statement in a <BODY> tag, as follows:

```
<BODY onLoad="window.defaultStatus='Set default text here...'">
```

Window Object Methods

The window object also exposes several methods for use in your scripts, and Table 9.3 lists these methods.

Table 9.3. Window object methods.

Property Description alert() Displays an Alert message box for the user clearTimeout() Cancels a timeout previously set with the setTimeout method close() Closes a window confirm() Displays a Confirm message box with OK and Cancel buttons to let the user make a decision open() Opens a new window prompt() Displays a Prompt dialog box with a message and an edit box that enables the user to input data setTimeout() Waits a specified amount of time before evaluating an expression

Frame Object

The frame object is a property of a window object, and it can be part of a frames array as well. Table 9.4 shows the properties of a frame.

Table 9.4. Frame object properties.

Property Description frames An array depicting all the frames in a window name The NAME attribute of the <FRAME> tag length The number of child frames within a frame parent A reference to the window or frame containing the current frameset self A reference to the current frame window A reference to the current frame

Frames also provide the methods shown in Table 9.5.

Table 9.5. Frame object methods.

Method Description clearTimeout() Cancels a timeout previously set with the setTimeout method setTimeout() Waits a specified amount of time before evaluating an expression

Document Object

A document object is a property of a window or frame object and embodies an HTML document. An HTML document is defined by the current URL and is composed of two sections:

- The *head*, which is defined by the HTML <HEAD> tag. The head contains information defining a document's title and absolute URL base (used for relative URL links within a document).
- The *body*, which is defined by the HTML <BODY> tag. The body defines the entire body of the document, including all other HTML elements for the document.

Note

JavaScript functions are usually placed within the <HEAD> tag to ensure that they are loaded before the body, and that a user won't be able to inhibit JavaScript activity before a script is ready to handle user input.

Document Object Properties

A document object provides many properties for manipulating document data and appearance, and these are listed in Table 9.6.

Table 9.6. Document properties.

Property Description alinkColor Represents the ALINK attribute anchors An array reflecting all the anchors in a document bgColor Represents the BGCOLOR attribute cookie Specifies a cookie fgColor Represents the TEXT attribute forms An array reflecting all the forms in a document lastModified Represents the date a document was last modified linkColor Represents the LINK attribute links An array reflecting all the links in a document location Represents the complete URL of a document referrer Represents the URL of the calling document title Represents the contents of the <TITLE> tag vlinkColor Represents the VLINK attribute

These four JavaScript objects are also properties of the document object:

- anchor
- form
- history
- link

Document Methods

In addition to the properties listed in Table 9.6, the document object also provides the methods shown in Table 9.7.

Table 9.7. Document methods.

Method Description clear() Clears all content from a document window. close() Closes an output stream and displays data sent to the document window using the write() or writeln() methods. open() Opens a stream to collect the output of write() or writeln() methods. write() Writes one or more HTML expressions to a document in the specified window. writeln() Writes one or more HTML expressions to a document in the specified window followed by a newline character.

Location Object

The location object represents a complete URL, and the properties of a location object represent the different

elements that make up a complete URL.

Location Object Properties

The location object provides the properties shown in Table 9.8 to provide information relevant to a location.

Table 9.8. Location object properties.

Property Description hash Defines an anchor name in the URL host Defines the *hostname:port* portion of the URL hostname Defines the host and domain name, or IP address, of a network host href Defines the entire URL pathname Defines the *url-path* portion of the URL port Defines the communications port that the server uses for communications protocol Defines the beginning of the URL, including the colon search Specifies a query

These properties are used as elements of a URL as follows:

```
protocol//hostname:port pathname search hash
```

History Object

A history object "remembers" where a user has been during the course of a browsing session. The history object has only one property: length. The length property reveals the number of entries in a history list. History lists are available for windows, frames, and documents, and Table 9.9 lists the methods available for this object.

Table 9.9. History object methods.

Method Description back Performs the same action as when a user clicks the Back button in a browser forward Performs the same action as when a user clicks the Forward button in a browser go Navigates to the location in the history list specified by a numeric parameter or a string parameter

Form Object

A form is a very important construct for JavaScript, and form *elements* enable users to interact with a document through familiar user interface controls. Each form in a document is a distinct object composed of various other objects including

- button
- checkbox
- hidden
- password
- radio
- reset
- select

- submit
- text
- textarea

Table 9.10 shows the properties available for the form object.

Table 9.10. Form object properties.

Property Description action Represents the ACTION attribute elements An array containing all the elements in a form encoding Represents the ENCTYPE attribute length Represents the number of elements on a form method Represents the METHOD attribute target Represents the TARGET attribute

A forms array has a single property, length, which reveals the number of forms in a document's forms array.

A form has a single method, submit, which sends data back to an HTTP server. The submit method returns data using one of two methods: get or post (specified by the method property).

A form also has a single event handler, onSubmit, which occurs when a user submits a form. This results in JavaScript code defined within the event handler being executed.

Text, Textarea, and Password Objects

A text object is a single-line text input box in an HTML form that enables a user to enter text or numbers. Similarly, a textArea object is a multiple-line text input box in an HTML form that enables a user enter text or numbers. A password object is a single-line text input box in an HTML form that enables a user enter text or numbers but masks the values entered by displaying asterisks (*).

Text, Textarea, and Password Properties

All three of these objects are closely related and have the same basic properties and methods. Table 9.11 shows the properties defined for these objects.

Table 9.11. Text object properties.

Property Description defaultValue Represents the text in an object name Represents the name of an object found in the NAME attribute value Represents the current value of the text object's data as set by the VALUE attribute

The defaultValue property is different for each of these objects, as follows:

- For a text object, defaultValue is initially the value of the VALUE attribute
- For a textarea object, defaultValue initially reflects the value specified between the <TEXTAREA> and </TEXTAREA> tags
- For a password object, defaultValue initially is null no matter what value is specified for the VALUE attribute

Text, Textarea, and Password Methods

Table 9.12 shows the methods defined for these objects.

Table 9.12. Text object methods.

Method Description focus() Gives a text object the input focus blur() Removes the focus from a text object select() Selects the input area of the specified text object

Text, Textarea, and Password Event Handlers

There are three event handlers that correspond to the three methods listed in Table 9.12 and one additional method that fires when the value of a text, textArea, or password object changes. Table 9.13 shows the event handlers defined for these objects.

Table 9.13. Text object event handlers.

Event Handler Indicates onBlur A text, textArea, or password object has lost the input focus onChange The data has changed in a text, textArea, or password object onFocus A text, textArea, or password object has received the input focus onSelect Text within a text, textArea, or password object has been selected

Hidden Object

A hidden object is a form element used to pass value/name pairs when a form's data is submitted. As such, a hidden object must be defined within a <FORM> tag.

A hidden object has only two properties:

- name, which represents the name of the object as specified by the NAME attribute.
- value, which represents the current value of the hidden object.

Button Object

A button object is a pushbutton on an HTML form. A button has two properties:

- The name of the button as specified by the NAME attribute.
- The text displayed on the button as specified by the VALUE attribute.

A button has one method, click(), which simulates a mouse click on the button. A button also has a corresponding event handler, onClick, that fires when the button is clicked.

Radio Object

A radio object is an array of mutually exclusive radio buttons on an HTML form that enable a user to choose one item from a list. The radio object has a click() method and an onClick event handler, just like a button object. In addition, a radio object supports the properties listed in Table 9.14.

Table 9.14. Radio object properties.

Property Description checked Specifies that a radio button element is selected defaultChecked Specifies whether a radio button element is initially checked or not, as specified with the CHECKED attribute length The number of radio buttons in a radio object name The name of the radio object as specified by the NAME attribute value The text of the radio object as specified by the VALUE attribute

Note

All radio buttons in a group must be referenced by their index numbers within the radio button array.

Checkbox Object

A checkbox object is an HTML form element that can be toggled to checked or not checked (on or off). A checkbox object has a click() method and an onClick event handler, just like a button object. In addition, a checkbox object supports the properties listed in Table 9.15.

Table 9.15. Checkbox object properties.

Property Description checked Specifies that a checkbox is checked defaultChecked Specifies whether the checkbox is initially checked or not as specified with the CHECKED attribute name The name of the checkbox object as specified by the NAME attribute value The text of the checkbox object as specified by the VALUE attribute

Select Object

A select object is an HTML form element that can take two forms:

- A *selection list* on an HTML form, in which case only one element in the list can be selected at a time (analogous to a drop-down list box in a Windows program).
- A scrolling list on an HTML form, in which case a user can choose one or more items from a list simultaneously (analogous to a multiple-selection list box in a Windows program).

Select Object Properties

A select object has the properties listed in Table 9.16.

Table 9.16. Select object properties.

Property Description length The number of options in a select object name The name of the object as specified by the NAME attribute options An array of options a user can choose from as specified by the <OPTION> tag selectedIndex The index of the selected option (or the first selected option if multiple options are selected)

The Options Array

The options array contains an entry for each option in a select object, and the select object's options array itself has several properties. These properties are listed in Table 9.17.

Table 9.17. The properties for an options array.

Property Description defaultSelected The element that's selected by default, as specified by the SELECTED attribute index The index of an option length The number of options in a select object name The name of an options element as specified by the NAME attribute selected A Boolean value specifying the current selection state of an option element selectedIndex The index of the selected option text The text following an <OPTION> tag that's displayed for an option element value The value of the name of an option element as specified by the VALUE attribute

In addition, select objects use the blur() and focus() methods described earlier, and the corresponding onBlur and onFocus event handlers, as well as the onChange event handler.

Submit Object

A submit object is a special button on an HTML form that causes a form's data to be submitted. A submit object supports the name and value properties discussed previously, as well as the click() method and the corresponding onClick event.

Link Object

A link object is some text or image that acts as a hyperlink. When the link object is clicked, the link reference is loaded into its target window.

Note

A document stores link objects in an array that can be referenced by a link element. The length property reflects the number of links in a document.

Link Object Properties

A link object has several properties as listed in Table 9.18.

Table 9.18. Link object properties.

Property Description hash A string beginning with a hash mark (#) that specifies an anchor name in a URL host Specifies the *hostname:port* element of a URL hostname Specifies the host and domain name, or IP address, of a network host href Specifies the entire URL. pathname Specifies the *url-path* portion of the URL port Specifies the communications port that the server uses for communications protocol Specifies the beginning of the URL, including the colon search Specifies a search query target Represents the TARGET attribute (see the section "Anchor Object")

Link Object Event Handlers

Link objects support two event handlers:

- onClick, which occurs in response to the link being clicked.
- onMouseOver, which occurs when the mouse moves over a link.

Anchor Object

An anchor object is some text that can be the target of a link object. Anchor objects use the HTML <A> tag, and a document stores anchors in an array that can be referenced by an element index. A simple example of an anchor is as follows:

```
<A NAME="SomeAnchorName" SomeAnchorText </A>
```

In this example, the NAME attribute specifies a name called *SomeAnchorName* that can be jumped to from a link within the current document. The text *SomeAnchorText* specifies the text to display at the anchor. To jump to this anchor from somewhere within a document, you can use the following simple link object:

```
<A HREF=SomeAnchorName Go to the SomeAnchorName anchor.</A>
```

Built-In Objects

Built-in JavaScript objects don't have visible user interfaces, and they don't use event handlers. They are used within an HTML page to provide ancillary services for your scripts and are restricted to properties and methods. Three built-in objects are discussed in this chapter:

- Date Object
- String Object
- Math Object

Date Object

The date object represents a date and has several useful methods available for working with dates within a document.

Date Object Methods

The date object methods are listed in Table 9.19.

Table 9.19. Date object methods.

Method Description getDate Returns the day of the month for the specified date (an integer, 1 through 31) getDay Returns the day of the week for the specified date (an integer, 0 through 6) getHours Returns the hour for the specified date (an integer, 0 through 23) getMinutes Returns the minutes in the specified date (an integer, 0 through 59) getMonth Returns the month in the specified date (an integer, 0 through 11) getSeconds Returns the seconds in the specified time (an integer, 0 through 59) getTime Returns the numeric value (in milliseconds since January 1, 1970) corresponding to the time for the specified date getTimezoneOffset Returns the time zone offset (between local and GMT time) in minutes for the current locale getYear Returns the year in the specified date parse Returns the number of milliseconds in a date string (since January 1, 1970) setDate Sets the day of the month for the specified date (an integer, 1 through 31) setHours Sets the hours for a specified date (an integer, 0 through 23) setMinutes Sets the minutes for a specified date (an integer, 0 through 59) setMonth Sets the month for a specified date (an integer, 0 through 11) setSeconds Sets the seconds for a specified time (an integer, 0 through 59) setTime Sets the number of milliseconds since January 1, 1970 in the time for a specified date setYear Sets the year for a specified date toGMTString Converts a date to a string, using Internet GMT conventions toLocaleString Converts a date to a string, using a user's system locale conventions UTC Returns the number of milliseconds in a date object (since January 1, 1970) in Universal Coordinated Time (UTC)

String Object

A string object represents a string of characters. The string object has only one property, length, which returns the number of characters in the string.

String Object Methods

The string object provides a wide variety of methods for use in manipulating strings, and these are listed in Table 9.20.

Table 9.20. String methods.

Method Description anchor() Used with the document write() or writeln() methods to create and display programmatically an anchor in a document big() Causes the specified text to be displayed in a big font, just as it would appear using the HTML <BIG> tag blink() Causes the specified text to blink, just as the HTML <BLINK> tag does bold() Causes a string to be displayed as bold text, just as the HTML <B> tag does charAt() Returns the character at the specified index in the string's array of characters fixed() Causes a string to be displayed with a monospace font, just as the HTML <TT> tag does fontcolor() Causes a string to be displayed in a specified color,

just as the HTML <FONT COLOR=*color*> tag does fontsize() Causes a string to be displayed in the specified font size, just as the HTML <FONTSIZE=*size*> tag does indexOf() Returns the index of the first occurrence of a character that matches a specified search string italics() Causes a string to be displayed as italic text, just as the HTML <I> tag does lastIndexOf() Returns the index of the last occurrence of a character that matches a specified search string link() Creates a hypertext link to jump to another URL. small() Causes a string to be displayed with a small font, just as the HTML <SMALL> tag does strike() Causes a string to be displayed as struck-out text, just as the HTML <STRIKE> tag does sub() Causes a string to be displayed as subscript, just as the HTML <SUB> tag does substring() Returns a substring from within a given string object sup() Causes a string to be displayed as superscript, just as the HTML <SUP> tag does toLowerCase() Converts a string to all lowercase letters toUpperCase() Converts a string to all uppercase letters

Math Object

The Math object has built-in properties and methods for mathematical constants and functions. This makes the use of complex mathematical equations easy in JavaScript! The following sections examine the properties and methods provided by this useful object.

Math Object Properties

The math object properties are actually just common, precalculated mathematical constants. These properties are very useful when performing complex calculations and are described in Table 9.21.

Table 9.21. Math object properties.

Property Description E Euler's constant and the base of natural logarithms (about 2.718) LN2 The natural logarithm of two (about 0.693) LN10 The natural logarithm of ten (about 2.302) LOG2E The base 2 logarithm of e (about 1.442) LOG10E The base 10 logarithm of e (about 0.434) PI The ratio of the circumference of a circle to its diameter (about 3.14159) SQRT1_2 The square root of 1/2, or, one over the square root of two (about 0.707) SQRT2 The square root of two (about 1.414)

Math Object Methods

The Math object methods are useful mathematical functions, most with equivalents in the C runtime library. Table 9.22 lists these methods.

Table 9.22. Math object methods.

Method Description abs() Calculates the absolute value of a number acos() Calculates the arc cosine (in radians) of a number asin() Calculates the arc sine (in radians) of a number atan() Calculates the arc tangent (in radians) of a number ceil() Returns the smallest integer greater than or equal to a number cos() Calculates the cosine of a number exp() Returns Euler's constant to the *nn* power, where *nn* is a number supplied as a parameter. floor() Returns the smallest integer less than or equal to a number log() Calculates the natural logarithm (base e) of a number max() Returns the larger of two numbers min() Returns the smaller of two numbers pow() Calculates a base number to the exponent power random() Returns a pseudo-random number between zero and one round() Rounds a number to the nearest integer sin() Calculates the sine of a number sqrt() Calculates the square root of a

number tan() Calculates the tangent of a number

Note

The random() method is available on UNIX platforms only.

Now that you've looked at the objects, properties, methods, and events offered by JavaScript, take a look at some examples in HTML code.

Creating JavaScript in HTML

A JavaScript applet is created using the HTML <SCRIPT> tag and by specifying the language to be JavaScript. Everything within the script block is part of the JavaScript, but standard HTML comments within a script block can be used to hide the script from browsers that don't support JavaScript. For example, the following HTML code uses a minimal script that displays the string "JavaScript rocks!" on the page box if the browser supports JavaScript:

```
<HTML>

<HEAD>

<TITLE>JavaScript Example 1</TITLE>

</HEAD>

<BODY>

<H1>JavaScript Example 1</H1><HR>

<SCRIPT LANGUAGE="JavaScript">

   document.write("JavaScript rocks!!")

</SCRIPT>

</BODY>

</HTML>
```

Figure 9.1 shows the result of the script in Internet Explorer 3.0.

Figure 9.1. A simple JavaScript in Internet Explorer 3.0.

This is fine for JavaScript-capable browsers, but browsers that don't understand JavaScript will simply display the JavaScript code in the page, as the CompuServe Mosaic browser does in Figure 9.2.

Figure 9.2. A script in a browser that doesn't understand JavaScript.

Because some users will undoubtedly use browsers that don't support JavaScript, you can prevent the script from appearing at all in these browsers by enclosing the entire script code in HTML comments. The revised code from

number tan() Calculates the tangent of a number

Note

The random() method is available on UNIX platforms only.

Now that you've looked at the objects, properties, methods, and events offered by JavaScript, take a look at some examples in HTML code.

Creating JavaScript in HTML

A JavaScript applet is created using the HTML <SCRIPT> tag and by specifying the language to be JavaScript. Everything within the script block is part of the JavaScript, but standard HTML comments within a script block can be used to hide the script from browsers that don't support JavaScript. For example, the following HTML code uses a minimal script that displays the string "JavaScript rocks!" on the page box if the browser supports JavaScript:

```
<HTML>

<HEAD>

<TITLE>JavaScript Example 1</TITLE>

</HEAD>

<BODY>

<H1>JavaScript Example 1</H1><HR>

<SCRIPT LANGUAGE="JavaScript">

   document.write("JavaScript rocks!!")

</SCRIPT>

</BODY>

</HTML>
```

Figure 9.1 shows the result of the script in Internet Explorer 3.0.

Figure 9.1. A simple JavaScript in Internet Explorer 3.0.

This is fine for JavaScript-capable browsers, but browsers that don't understand JavaScript will simply display the JavaScript code in the page, as the CompuServe Mosaic browser does in Figure 9.2.

Figure 9.2. A script in a browser that doesn't understand JavaScript.

Because some users will undoubtedly use browsers that don't support JavaScript, you can prevent the script from appearing at all in these browsers by enclosing the entire script code in HTML comments. The revised code from

number tan() Calculates the tangent of a number

Note

The random() method is available on UNIX platforms only.

Now that you've looked at the objects, properties, methods, and events offered by JavaScript, take a look at some examples in HTML code.

Creating JavaScript in HTML

A JavaScript applet is created using the HTML <SCRIPT> tag and by specifying the language to be JavaScript. Everything within the script block is part of the JavaScript, but standard HTML comments within a script block can be used to hide the script from browsers that don't support JavaScript. For example, the following HTML code uses a minimal script that displays the string "JavaScript rocks!" on the page box if the browser supports JavaScript:

```
<HTML>

<HEAD>

<TITLE>JavaScript Example 1</TITLE>

</HEAD>

<BODY>

<H1>JavaScript Example 1</H1><HR>

<SCRIPT LANGUAGE="JavaScript">

   document.write("JavaScript rocks!!")

</SCRIPT>

</BODY>

</HTML>
```

Figure 9.1 shows the result of the script in Internet Explorer 3.0.

Figure 9.1. A simple JavaScript in Internet Explorer 3.0.

This is fine for JavaScript-capable browsers, but browsers that don't understand JavaScript will simply display the JavaScript code in the page, as the CompuServe Mosaic browser does in Figure 9.2.

Figure 9.2. A script in a browser that doesn't understand JavaScript.

Because some users will undoubtedly use browsers that don't support JavaScript, you can prevent the script from appearing at all in these browsers by enclosing the entire script code in HTML comments. The revised code from

the preceding example would give you this:

```
<HTML>

<HEAD>

<TITLE>JavaScript Example 1</TITLE>

</HEAD>

<BODY>

<H1>JavaScript Example 1</H1><HR>

<SCRIPT LANGUAGE="JavaScript">

<!-- Hide JavaScript from old browsers

   document.write("JavaScript rocks!!")

// End code hiding -->

</SCRIPT>

</BODY>

</HTML>
```

JavaScript Functions

The real power of JavaScript begins to show when you add functions to the mix. As any modern programming language should, JavaScript supports the notion of functions that live in one area and can be called from elsewhere in the code. Where do JavaScript functions live in HTML? Typically in the <HEAD> section. This is because the function code will be loaded and interpreted before the page is displayed, and any JavaScript source that calls a function will have it available instantly.

A JavaScript function is defined (appropriately) by the keyword function, and it must fall within a <SCRIPT> block. The function is given a name, and this name can be called from a JavaScript code. Take a look at a function called displayMessage(), which takes a typeless parameter called msg and sends it to the JavaScript alert() method.

```
function displayMessage(msg)

{

   alert(msg)

}
```

This function could then be called by placing a form pushbutton on the page and calling the dislayMessage() function in response to the button's onClick event, like this:

```
<FORM>

   <INPUT TYPE="button"

      VALUE="Click Here..."

      onClick="displayMessage('Test message...')">

</FORM>
```

Note

The string Test message... is surrounded with single quotes. This is because they are nested within the double quotes used for the displayMessage() function call by the onClick event.

The complete source code for this function example is given in Listing 9.1.

Listing 9.1. Calling a function from within a JavaScript.

```
<HTML>

<HEAD>

<TITLE>JavaScript Example 2</TITLE>

<SCRIPT LANGUAGE="JavaScript">

<!--

function displayMessage(msg)

{

   alert(msg)

}

-->

</SCRIPT>

</HEAD>

<BODY>

<H1>JavaScript Example 2</H1><HR>

This script calls a simple function...click the button!
```

```
<FORM>

   <INPUT TYPE="button"

      VALUE="Click Here..."

      onClick="displayMessage('Test message...')">

</FORM>

</BODY>

</HTML>
```

In this example, the string "Test message..." is sent to the function displayMessage(), causing an alert box to appear. Now extend this into a more useful example by enabling a user to specify the text displayed in the alert box.

For this example, a text object named text1 is used to enable the user to input the text to display in the alert box. The showMessage() function is modified here to take a form object as a parameter, like this:

```
<SCRIPT LANGUAGE="JavaScript">

<!-- Updated showMessage() function takes a form as a parameter

function showMessage(form)

{

   if (form.text1.value.length > 0)

      alert("You entered the text: " + form.text1.value)

   else

      alert("You must enter some text!")

}

-->

</SCRIPT>
```

Notice that if the length of the text isn't greater than zero, the script tells the user to enter some text; otherwise, the text is displayed using the form.text1.value property to get the string.

To enhance the page even more, the example uses a date object to extract information about when the page was last modified. This information is then written to the bottom of the page to provide automatic update information whenever the page changes, like this:

```
<H5>

<SCRIPT LANGUAGE="JavaScript">
```

```
<!-- Hide script from old browsers

   update = new Date(document.lastModified)

   month  = update.getMonth() + 1

   date   = update.getDate()

   year   = update.getYear()

   document.writeln("<I>Last modified: " +

      month + "/" + date + "/" + year + "</I>")

// End script hiding -->

</SCRIPT>

</H5>
```

The complete code listing for this example is given in Listing 9.2.

Listing 9.2. A page that gathers text data from a simple form and displays the document's last modified date.

```
<HTML>

<HEAD>

<TITLE>JavaScript Example 3</TITLE>

<SCRIPT LANGUAGE="JavaScript">

<!-- Updated showMessage() function takes a form as a parameter

function showMessage(form)

{

   if (form.text1.value.length > 0)

      alert("You entered the text: " + form.text1.value)

   else

      alert("You must enter some text!")

}

-->

</SCRIPT>

</HEAD>
```

```
<BODY>

<H1>JavaScript Example 3</H1>

<HR>

This script uses the data you enter in the text area and

 displays it in an alert box...click the button!

<!-- Start a form for getting the text to use -->

<FORM>

   <INPUT TYPE="button" VALUE="Click Here..."

      onClick="showMessage(this.form)">

   <INPUT TYPE="text" SIZE=40 MAXLENGTH=256 NAME="text1"></p>

</FORM>

<!-- Show the last modified date in the document with this simple script -->

<H5>

<SCRIPT LANGUAGE="JavaScript">

<!-- Hide script from old browsers

   update = new Date(document.lastModified)

   month  = update.getMonth() + 1

   date   = update.getDate()

   year   = update.getYear()

   document.writeln("<I>Last modified: " +

      month + "/" + date + "/" + year + "</I>")

// End script hiding -->

</SCRIPT>

</H5>

</BODY>

</HTML>
```

The results of this example can be seen in Figure 9.3, with the text box value displayed in an alert box, and the date the document was last modified displayed as part of the document.

Figure 9.3. EX3.HTM as it appears in Internet Explorer 3.0.

These few small examples should give you an idea of the power and ease of JavaScript. In Chapter 12, "Advanced Web Page Creation," you'll see some more useful and exciting examples for working with JavaScript and frames.

Summary

JavaScript provides a powerful way for Web page authors to add interactivity to Web pages with a well-structured scripting language. Programmers familiar with C, C++, or Java should have no trouble learning it. When combined with other tools and some imagination, JavaScript can provide the means to create a truly exciting Web browsing experience for the user.

The next chapter examines Microsoft's Web authoring tool FrontPage, a versatile and all-around excellent package for designing and maintaining Web sites.

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☑ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☑ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.